Implementing Policy-Based Routing

Policy-Based Routing

Table 1: Feature History Table

Feature Name	Release Information	Feature Description

Policy-Based Routing byte count	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100, P200], 8700 [ASIC: P100, K100], 8010 [ASIC: A100])(select variants only); Modular Systems (8800 [LC ASIC: P100])(select variants only*) The bytes count enables the router to accurately track and display the total number of bytes matched by each Policy-Based Routing (PBR) rule. This provides enhanced visibility into traffic volume, allowing you to better monitor and analyze network usage. The byte count in the **show pbr stats counters policy <policy_name>** command now supports additional PIDs. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 8011-4G24Y4H-I • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 8223-64EF-M(O)
Policy-Based Routing	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A • 8011-12G12X4Y-D
Policy-Based Routing	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Policy-Based Routing	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on Cisco 8712-MOD-M routers.

Policy-Based Routing	Release 24.2.11	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) You can now create customised routing policies based on different parameters such as IP address, port numbers, or protocols. With Policy-Based Routing (PBR), you can enhance your network security by steering sensitive data away from potentially vulnerable network segments. Also, by allowing you to distribute traffic across multiple paths, PBR can help prevent traffic congestion in your network. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Policy-Based Routing (PBR) gives you a flexible means of routing packets by allowing you to configure a defined policy for traffic flows, reducing reliability on routes derived from routing protocols. Moreover, PBR allows you to prioritize and provide a specific routing path for certain types of traffic (for example, VoIP, video conferencing) based on the service-level agreement (SLA).

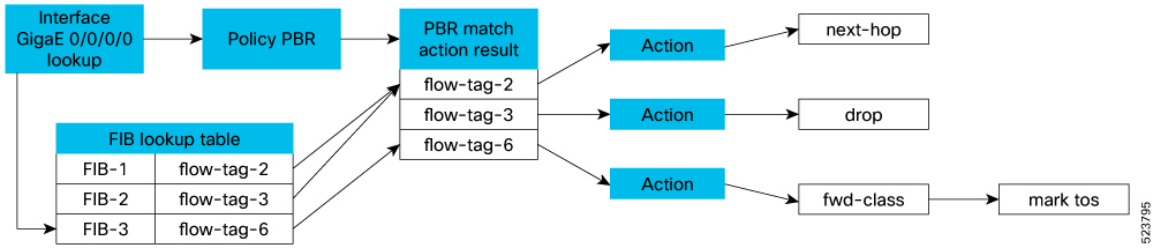
Unlike traditional routing which is based on destination IP address alone, PBR allows you to route packets based on different parameters such as IP address, port numbers, or protocols. For example, you can implement routing policies to allow or deny paths based on the identity of a particular end system, an application protocol, or the size of packets.

Flow-tag Support

A provider edge (PE) router directs traffic toward the core through the routes learnt. It also installs policies on your interfaces to offer value-added services based on your profiles. The administrators must manually scan route-tables and install traffic-path selection as Access Control Lists (ACLs) on your interfaces for proper traffic forwarding. However, with the scale of route prefix-lists learned from the core, the hardware resource requirements to program policies grows high. When route updates change, it requires administrators to manually update these policies. Moreover, these route policies are applied per prefix list and do not allow per interface selection to associate these policies on a per customer basis, which can be a challenge.

To simplify policy management and improve scalability, you can classify the traffic early in the routing domain as routes are learned, and mark them with metadata or a tag in the Forwarding Information Base (FIB). Then, forwarding-path rules can be defined against this flow-tag value. This could reduce the need for manual updates and make the system more scalable and efficient.

The following figure illustrates the process:



Flow-tag is set and distributed via the Routing Information Base (RIB) as a policy attribute of the Forwarding Information Base (FIB) entry in the FIB lookup table. Route Policy Language (RPL) is used to create these flow tags through the "set" operation. The flow tag is then referred to in the PBR policy, where it is associated with specific actions or policy rules against the flow tag's value.

Forward-Class Support

The PBR class-map defines the matching criteria for classifying a particular type of traffic, while the forward-class defines the forwarding path these packets must take. Once a class-map is associated with a forwarding-class in the policy map, all the packets that match the class-map are forwarded as defined in the policy-map.

Traffic Engineering (TE) tunnels are then used to direct traffic along specific paths through the network. These TE tunnels can be associated with a forward-class which determines the forwarding path for the packets.

The use of the **auto-route** command allows the TE interface to be exported to the routing protocol module, associating the route in the Forwarding Information Base (FIB) database with these tunnels.

The system can support up to eight forward-classes with eight TE tunnels each, allowing a maximum of 32 TE tunnels to be associated with the destination route.

Guideline for forward-class implementation

When you configure `match tcp-flag`, also configure `match protocol tcp` to ensure correct traffic classification.

Access-Group Support

PBR supports matching on an ACL group ID. You can specify large prefix lists and long lists of specific permit/deny lists. The prefix lists serve as a filter for routing updates, and the permit or deny lists determine what action to take when a match is found.

Permit/Deny Actions in ACL Groups

An access control entry (ACE) can be either a HIT or a MISS. When the ACE is a HIT, the following actions are performed:

- Permit - Stop processing ACL, and if
  - the type of class is “match-any”, proceed to take policy-map action
  - the type of class is “match-all”, proceed to the next ACL or “match” statement in the class-map considering that you already have a match (true) for an existing ACL.
- Deny - Stop processing ACL and if
  - the type of class is “match-any”, proceed to the next ACL (or “match”) statement in the class-map if there is no match for the existing ACL.

- the type of class is “match-all”, exit the class. No policy action is taken, and the next class is processed as per the specified order.

When the ACE is a MISS (didn’t match any permit or deny statement), the following actions are performed:

- Proceed to the next "match" statement.
- If the “match” statement does not provide a match, then skip that class. No policy action is taken for that class, and proceed to check the packet against the next class in order, or proceed to L3 forwarding lookup.

Note An explicit "deny" entry is not supported in ACL groups that are embedded in class-maps.

Supported Match and Set Operations

The following table illustrates the match/set criteria that is supported by PBR:

Table 2: Supported Match and Set Operations

Criteria	match/set
source ip	match
destination ip	match
source protocol/port	match
destination protocol/port	match
nexthop ip	set
nexthop vrf	set
nexthop ip+vrf	set
forward-class	set
access-group	match
flow-tag	match
ip protocol	match
tcp-flag	match
port-range	match

Restrictions for Implementing Policy-Based Routing

The following are the restrictions for implementing Policy-based routing:

- QoS Group and Flow-tag are not supported together at the same time.

- Bridge Group Virtual Interface (BVI) and Pseudowire Headend (PHWE) subinterfaces supports PBR from Release 26.1.1.
- BGP Flowspec feature and PBR are not supported together on the same interface.
- A route-policy can have either 'set qos-group' or 'set flow-tag,' but not both for a prefix-set.
- Route policy for qos-group and route policy flow-tag cannot have overlapping routes. The Quality-of-service Policy Propagation Using Border Gateway Protocol (QPPB) and flow tag features can coexist (on same as well as on different interfaces) as long as the route policy used by them do not have any overlapping route.
- Mixing usage of qos-group and flow-tag in route-policy and policy-map is not recommended.

Configure Policy-Based Routing

PBR configuration includes the following steps:

1. Configure Flow-tag
2. Provision Forward Class using RPL
3. Configure ACLs with Policy-Based Routing

Configure Flow-tag

The following is a sample definition of a named AS path set in the route policy:

```
as-path-set as-set-1
   ios-regex '_12$',
   ios-regex '_13$'
end-set
```

Use the following sample configurations to set the Flow-tag, and apply the related configuration.

```
/* Set the flow-tag under route-policy configuration */

Router(config)# route-policy flowtag_match
Router(config-rpl)# if community matches-every (100:1) then
   set flow-tag 31
else
Router(config-rpl-else)# if as-path in as-set-1 then
   set flow-tag 62
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy

/* Apply the policy when updating the routing table */

Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 209.165.201.19
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy flowtag_match

/* Configure a class map */
```

```
Router(config)# class-map type traffic match-all green-tag1
Router(config-cmap)# match flow-tag 31
Router(config-cmap)# end-class-map
Router(config-cmap)# exit
Router(config)# policy-map type pbr nh_select
Router(config-pmap)# class type traffic green-tag1
Router(config-pmap-c)# set forward-class 1

/* Configure an interface and apply the PBR policy map to the interface */

Router(config)# interface HundredGigE0/7/0/27
Router(config-if)# ipv4 address 10.10.20.10
Router(config-if)# service-policy type pbr input nh_select
```

Running Configuration

```
Router# show running-config class-map
class-map type traffic match-all green-tag1
match flow-tag 101
 end-class-map
!


Router# show running-config policy-map
policy-map type pbr nh_select
class type traffic green-tag1
  redirect ipv4 nexthop 10.1.2.2
 !
 class type traffic green-tag1
 !
 end-policy-map
!


Router# show running-config interface HundredGigE0/7/0/27
interface HundredGigE0/7/0/27
service-policy type pbr input nh_select
!
```

Provision Forward Class using RPL

Use the following sample configuration to provision Forward Class using RPL.

```
/* Set the forward class ID for community string */

Router(config)# route-policy c1
Router(config-rpl)# if community matches-every (6500:1) then
   set forward-class 1
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy

Router(config)# router bgp 50
Router(config-bgp)# bgp router-id 209.165.201.29
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy c1
Router(config-bgp-af)# exit
Router(config-bgp)# exit
```

```
Router(config)# interface tunnel-te1
Router(config-if)# forward-class 1
```

In this example, BGP on the receiving PE is configured with a table policy, and all routes that matches the community string are tagged with a forward-class-id of 1.

```
/* Set the forward class ID for VRF */

Router(config)# route-policy c1
Router(config-rpl)# set forward-class 1
Router(config-rpl)# end-policy

Router(config)# route-policy c2
Router(config-rpl)# set forward-class 2
Router(config-rpl)# end-policy

Router(config)# router bgp 50
Router(config-bgp)# bgp router-id 209.165.201.29
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# exit

Router(config-bgp)# vrf one
Router(config-bgp-vrf)# rd 1:1
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy c1
Router(config-bgp-af)# exit
Router(config-bgp)# exit

Router(config-bgp)# vrf two
Router(config-bgp-vrf)# rd 2:2
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy c2
Router(config-bgp-af)# exit
Router(config-bgp)# exit

Router(config)# interface tunnel-te1
Router(config-if)# forward-class 1

Router(config)# interface tunnel-te2
Router(config-if)# forward-class 2
```

In this example, BGP on receiving PE is configured with a table-policy (C1) and (C2) for two different VRFs.

The policy (C1) sets forward-class 1. From the available TE paths, tunnel-te1 with forward-class 1 is selected for forwarding. Similarly for VRF two, traffic tunnel-te2 associated with forward-class 2 is selected for forwarding.

```
/*Configure a route policy with next hop and set a forward class */

Router(config)# prefix-set nh-set-1
Router(config-pfx)# 10.10.0.1
Router(config-pfx)# end-set

Router(config)# route-policy c1
Router(config-rpl)# if next-hop in nh-set-1 then
```

```
   set forward-class 1
Router(config-rpl)# endif
Router(config-rpl)# end-policy

Router(config)# router bgp 50
Router(config-bgp)# bgp router-id 209.165.201.29
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy c1
Router(config-bgp-af)# exit
Router(config-bgp)# exit

Router(config)# interface tunnel-te1
Router(config-if)# forward-class 1
```

In this example, BGP on receiving PE is configured with a table-policy (C1), and the policy (C1) sets the forward-class. From the available TE paths, tunnel-te1 with forward-class 1 is selected for forwarding.

Configure ACLs with Policy-Based Routing

Use the following sample configuration to configure ACLs with PBR.

```
/* Configure an access list */
Router(config)# ipv4 access-list INBOUND-ACL
Router(config-ipv4-acl)# 10 permit ipv4 any host 10.1.1.10
Router(config-ipv4-acl)# 20 permit ipv4 any host 10.2.3.4
Router(config-ipv4-acl)# commit
Mon Nov  6 17:22:42.529 IST
Router(config-ipv4-acl)# exit

/* Configure a class map for the access list */
Router(config)# class-map type traffic match-any INBOUND-CLASS
Router(config-cmap)# match access-group ipv4 INBOUND-ACL
Router(config-cmap)# end-class-map
Router(config)# commit
Mon Nov  6 17:29:12.026 IST

/* Configure a PBR policy map with the class map */
Router(config)# policy-map type pbr INBOUND-POLICY
Router(config-pmap)# class type traffic INBOUND-CLASS
Router(config-pmap-c)# redirect ipv4 nexthop 192.168.10.1
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic class-default
Router(config-pmap-c)# transmit
Router(config-pmap-c)# commit
Mon Nov  6 17:25:33.858 IST
Router(config-pmap)# end-policy-map

/* Configure a GigE interface and apply the PBR policy map to the interface */
Router(config)# interface GigabitEthernet 0/0/0/0
Router(config-if)# ipv4 address 10.10.10.1 255.255.255.0
Router(config-if)# service-policy type pbr input INBOUND-POLICY
Router(config-if)# commit
Mon Nov  6 17:31:23.645 IST
Router(config-if)# exit
```

Running Configuration

```
Router# show running-config ip access-list
ipv4 access-list INBOUND-ACL
10 permit ipv4 host 10.10.10.1 any
!
```

ePBR drop and transmit actions

ePBR drop and transmit actions is a routing feature that

- provides granular control over how packets matching specific criteria in a PBR policy are handled
- discards unwanted or malicious traffic directly at the interface to improve security and filtering
- transmits selected traffic, bypassing PBR rules and following the standard routing table lookup, and
- simplifies policy design and improves operational clarity with dedicated actions for common traffic management needs.

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
ePBR drop and transmit actions	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on Cisco 8711-48Z-M routers.

Feature Name	Release Information	Feature Description
ePBR drop and transmit actions	Release 25.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100], 8010 [ASIC: A100]); Modular Systems (8800 [LC ASIC: P100]) The feature adds two critical forwarding actions, such as `drop` and `transmit` to enhanced Policy-Based Routing (ePBR) policies, giving network administrators precise control over how traffic matching specific criteria in a PBR policy is handled. These actions simplify policy creation, remove complex workarounds, and allow administrators to manage exceptions or security scenarios with granular per-traffic control. **CLI:** • The **drop** and **transmit** commands are introduced in the policy-map.

Drop and transmit actions for enhanced PBR control

The feature enhances ePBR by introducing two new forwarding actions, such as drop and transmit. These actions give network administrators explicit control over the final disposition of traffic matching specific PBR criteria.

Drop action: Discards unwanted or malicious traffic directly at the ingress interface, such as malicious traffic flows, unauthorized protocols, or unwanted sources directly at the ingress interface. This early filtering removes the need for complex ACL configurations, and prevents harmful traffic from consuming network resources.

Transmit action: Forwards selected traffic normally, bypassing PBR-specific next-hop or policy redirection rules, and follows the standard FIB lookup. This ensures that the packet follows the standard routing table lookup and is treated like regular traffic.

These actions complement the existing `redirect` and `set forward-class` actions, providing more comprehensive and flexible traffic steering capabilities. By making these actions explicit within ePBR policies, network administrators avoid relying on implicit defaults, complex null interface configurations, or intricate policy ordering.

This capability delivers precise per-traffic handling, strengthens network security, preserves service integrity for critical flows, clarifies operational intent, reduces configuration errors, and streamlines policy design.

Benefits of ePBR drop and transmit actions

ePBR drop and transmit actions offer these benefits:

- Ensures granular security and filtering for precise traffic handling decisions.
- Enhances network security by dropping malicious or irrelevant packets early.
- Flexible exception handling through the transmit option.
- Simplifies ePBR policies.

Enable drop or transmit actions for PBR policies

Follow these steps to enable drop or transmit actions for PBR policies.

Procedure

Step 1 Run the `class-map type` command to define class maps for the traffic you want to control.

Class maps define the match criteria for the PBR policy.

Example:

```
/* Configure class map to drop traffic */
Router(config)#class-map type traffic match-all UNWANTED-SOURCE-TRAFFIC
Router(config-cmap)#match source-address ipv4 192.168.1.0/24
Router(config-cmap)#end-class-map

/* Configure class map to transmit traffic */
Router(config)#class-map type traffic match-all MANAGEMENT-TRAFFIC
Router(config-cmap)#match destination-port 22
Router(config-cmap)#match destination-port 23
Router(config-cmap)#end-class-map
```

Step 2 Run the `policy-map type` command to create a policy map of type PBR, and assign the new actions to the defined classes.

Example:

```
Router(config)#policy-map type pbr INGRESS-SECURITY-POLICY

/* Assign the drop action to the UNWANTED-SOURCE-TRAFFIC class */
Router(config-pmap)#class type traffic UNWANTED-SOURCE-TRAFFIC
Router(config-pmap-c)#drop
Router(config-pmap-c)#exit

/* Assign the transmit action to the MANAGEMENT-TRAFFIC class */
Router(config-pmap)#class type traffic MANAGEMENT-TRAFFIC
Router(config-pmap-c)#transmit
Router(config-pmap-c)#exit
```

Step 3 (Optional) Run the `class type` command to define a class default action when no other class matches.

You can define a default action when traffic does not match any class, and it then follows normal routing.

Example:

```
Router(config-pmap)#class type traffic class-default
Router(config-pmap-c)#transmit
Router(config-pmap-c)#commit
Router(config-pmap)#end-policy-map
```

Step 4 Apply the defined policy map to the required interface.

Example:

```
Router(config)#interface HundredGigE 0/0/0/1
Router(config-if)#service-policy type pbr input INGRESS-SECURITY-POLICY
Router(config-if)#commit
Router(config-if)#exit
```

Step 5 Run the `show running-config` command to verify the configuration.

Example:

```
class-map type traffic match-all UNWANTED-SOURCE-TRAFFIC
 match source-address ipv4 192.168.1.0/24
end-class-map

class-map type traffic match-all MANAGEMENT-TRAFFIC
 match destination-port 22
 match destination-port 23
end-class-map

!
policy-map type pbr INGRESS-SECURITY-POLICY
 class type traffic UNWANTED-SOURCE-TRAFFIC
  drop
 !
 class type traffic MANAGEMENT-TRAFFIC
  transmit
 !
 class type traffic class-default
  transmit
 !
end-policy-map
!
interface HundredGigE 0/0/0/1
 service-policy type pbr input INGRESS-SECURITY-POLICY
!
```

Step 6 Run the `show controllers npu stats` command to verify the dropped packets.

Example:

In the sample output, the number of packets dropped is 9000.

```
Router#show controllers npu stats traps-all instance all location all | exclude "0
   0"
Tue Apr 15 19:44:11.681 PDT
…

Trap Type                                    NPU  Trap  Punt       Punt  Punt  Punt Configured
Hardware   Policer Avg-Pkt Packets             Packets
                                             ID   ID    Dest       VoQ   VLAN  TC  Rate(pps)
Rate(pps)  Level   Size    Accepted            Dropped
=============================================================================================
ONLINE_DIAG                                  0    32   RPLC_CPU    295   1538  7    8038660
8087158    IFG     64      362                 0
L3_ACL_DROP(D)                               0    106  NPU_DROP    ---   ---   ---  ----------
---------- ---     N/A     0                   9000
ISIS/L3                                     0    133  RPLC_CPU    295   1538  7    10000      9789
       IFG     1520    14                  0
```

ePBR on BVI

Enhanced Policy-Based Routing (ePBR) on Bridge Virtual Interface (BVI) is a routing feature that

- applies ingress security policies and traffic steering to Layer 2 traffic entering a Provider Edge (PE) router through a BVI,
- processes packets before they are routed to a Layer 3 interface, and
- provides granular control over traffic handling through dedicated redirect, drop, and transmit actions.

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
ePBR on BVI	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100]) (select variants only*); Centralized Systems (8400 [ASIC: K100]) (select variants only*); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) You can ensure secure and efficient traffic handling at the network ingress by applying ePBR policies directly to the BVI. This feature allows the Cisco IOS XR software to intercept and steer inbound Layer 2 traffic before it transitions to Layer 3 routing. *This feature is supported on: • 8212-48FH-M • 8404-SYS-D • 8711-32FH-M • 8711-48Z-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

The ePBR on BVI feature allows you to apply ingress security policies and traffic steering to Layer 2 traffic entering a PE router from a VPN through a BVI before it is routed to a Layer 3 interface.

By applying ePBR policies to a BVI, you can:

- **Redirect**: Forward traffic to a specific next-hop, bypassing the standard routing table.
- **Drop**: Discard malicious or unauthorized traffic at the ingress interface.
- **Transmit**: Explicitly permit traffic to follow standard routing table lookups.

For more information on ePRB, see ePBR drop and transmit actions, on page 10.

Limitations for ePBR on BVI

These are the limitations for ePBR on BVI. For more restriction about policy-based routing, see the Restrictions for Implementing Policy-Based Routing, on page 5.

- The policy supports only the ingress direction on the BVI.
- Supports upto 6k scale limits.

How ePBR on BVI works

In a PE environment, managing Layer 2 traffic from customer VPNs requires granular control before the traffic transitions to Layer 3 routing. Applying ePBR directly to the BVI allows the system to intercept and steer this traffic based on the requirements.

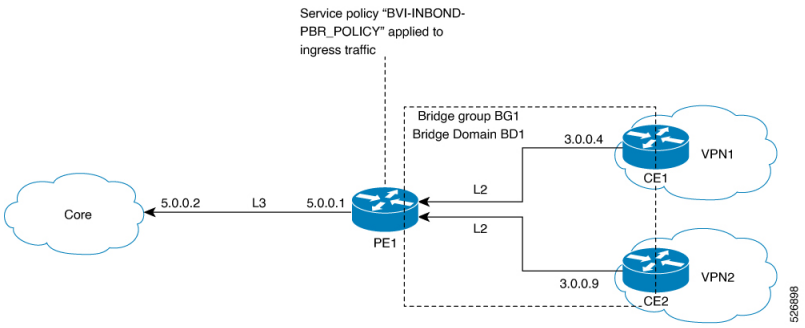
Summary

Consider a topology where the PE router connects CE devices to the core network using a BVI as the Layer 3 gateway.

- **CE1 and CE2**: Customer edge devices.
  - **CE1**: Source router in VPN1 with IP address 3.0.0.4.
  - **CE2**: Source router in VPN2 with IP address 3.0.0.9.
- **PE1**: Provider edge router.
- **Core**: Core router with IP address 5.0.0.2.
- **BVI20**: The routed interface with IP address 3.0.0.8 on PE1 that acts as the gateway for Bridge Group BG1 and Bridge Domain BD1.
- **BVI-INBOUND-PBR_POLICY**: The service policy "BVI-INBOUND-PBR_POLICY" is configured on the BVI20 interface on PE1 to define specific actions, such as transmit, drop, and redirect, which manage inbound traffic from the CEs before it reaches the core router.

Workflow

Figure 1: Network topology for ePBR on BVI



The Cisco IOS XR software takes the following steps to manage inbound traffic steering and filtering on a BVI using ePBR:

1. The CE devices CE1 or CE2 sends an inbound packet containing Ethernet, VLAN, and IP headers to the PE1 router through the Layer 2 bridge domain.
2. The Cisco IOS XR software receives the packet and directs it to the BVI20 interface for Layer 3 processing.
3. The Cisco IOS XR software evaluates the packet against the **BVI-INBOUND-PBR_POLICY** attached to the BVI.
4. Based on the policy match, the Cisco IOS XR software performs one of the following actions:

If the packet matches the...	Then the Cisco IOS XR software...
transmit class	transmits the packet normally using standard routing lookups.
drop class	discards the packet at the ingress interface.
redirect class	steers the packet to the core next-hop at IP 5.0.0.2.

5. If the packet does not match any specific class, the Cisco IOS XR software applies the **class-default** action and forwards the traffic according to the standard routing table.

Result

The ePBR process results in the automated filtering and steering of inbound Layer 2 traffic, ensuring that packets are handled according to security and traffic engineering policies before they reach the network core.

Configure ePBR on BVI

Perform the following steps to configure ePBR on BVI:

Procedure

Step 1 Configure the BVI interface on the local PE1 to establish the Layer 3 gateway for customer VPN traffic.

Example:

```
Router# configure
Router(config)# interface BVI20
Router(config-if)# ipv4 address 3.0.0.8 255.255.255.0
Router(config-if)# commit
```

Step 2 Configure the bridge domain and associate physical interfaces with the BVI to enable routing between the Layer 2 domain and the Layer 3 network.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface FourhundredGigE0/1/0/27
Router(config-l2vpn-bg-bd-ac)# interface FourhundredGigE0/0/0/8
```

```
Router(config-l2vpn-bg-bd-ac)# routed interface BVI20
Router(config-l2vpn-bg-bd-bvi)# commit
```

Step 3 Configure a point-to-point pseudowire connection with an xconnect group and neighbor specification on remote PE2 to establish a seamless Layer 2 VPN link between local PE1 and remote PE2.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group phy
Router(config-l2vpn-xc)# p2p p1
Router(config-l2vpn-xc-p2p)# interface FourhundredGigE0/0/0/7
Router(config-l2vpn-xc-p2p)# interface FourhundredGigE0/0/0/11.1
Router(config-l2vpn-xc-p2p)# commit
```

Step 4 Configure the traffic class or classmap for IPv4 and IPv6 to identify the traffic based on specific criteria.

Example:

The following is an example of classmap for IPv4:

```
Router(config)# class-map type traffic match-all ipv4_CM3
Router(config-cmap)# match destination-address ipv4 201.0.3.1 255.255.255.0
Router(config-cmap)# match source-address ipv4 192.1.1.0 255.255.255.0
Router(config-cmap)# match protocol tcp
Router(config-cmap)# match destination-port 1024
Router(config-cmap)# match source-port 1024
Router(config-cmap)# match tcp-flag 0x10
Router(config-cmap)# match access-group ipv4 ipv4_acl_3
Router(config-cmap)# match flow-tag 20
Router(config-cmap)# end-class-map
```

Example:

The following is an example of classmap for IPv6:

```
Router(config)# class-map type traffic match-all ipv6_CM1
Router(config-cmap)# match destination-address ipv6 2001:0:0:1::1/64
Router(config-cmap)# match source-address ipv6 1111::1:0/120
Router(config-cmap)# match protocol tcp
Router(config-cmap)# match destination-port 1024
Router(config-cmap)# match source-port 1024
Router(config-cmap)# match tcp-flag 0x10
Router(config-cmap)# match access-group ipv6 ipv6_acl
Router(config-cmap)# match flow-tag 10
Router(config-cmap)# end-class-map
```

Step 5 Create the ePBR policy-map for IPv4 and IPv6 to define the actions such as redirect, drop, or transmit for the classified traffic.

Example:

The following is an example of ePBR policy-map for IPv4:

```
Router(config)# policy-map type pbr BVI-INBOUND-PBR_POLICY
Router(config-pmap)# class type traffic ipv4_CM1
Router(config-pmap-c)# redirect ipv4 nexthop 192.0.1.2
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic ipv4_CM2
Router(config-pmap-c)# drop
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic ipv4_CM3
Router(config-pmap-c)# transmit
Router(config-pmap-c)# exit
```

```
Router(config-pmap)# class type traffic class-default
Router(config-pmap-c)# transmit
Router(config-pmap-c)# commit
Router(config-pmap)# end-policy-map
```

Example:

The following is an example of ePBR policy-map for IPv6:

```
Router(config)# policy-map type pbr BVI-INBOUND-PBR_POLICY
Router(config-pmap)# class type traffic ipv6_CM1
Router(config-pmap-c)# redirect ipv6 nexthop 2001:101::2
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic ipv6_CM2
Router(config-pmap-c)# drop
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic ipv6_CM3
Router(config-pmap-c)# transmit
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic class-default
Router(config-pmap-c)# transmit
Router(config-pmap-c)# commit
Router(config-pmap)# end-policy-map
```

Step 6 Attach the policy to the BVI interface to enforce traffic steering and security rules on inbound packets.

Example:

```
Router# configure
Router(config)# interface BVI20
Router(config-if)# service-policy type pbr input BVI-INBOUND-PBR_POLICY
Router(config-if)# commit
```

Step 7 Run these commands to verify the policies:

a) Run the **show interfaces BVI1 accounting** command to verify interface statistics.

Example:

```
Router# show interface BVI1 accounting
BVI1
  Protocol              Pkts In         Chars In     Pkts Out        Chars Out
  ARP                         0                0            2               84
  IPV6_ND                     0                0           35             3544
```

b) Run the **show controllers npu stats traps-all instance all location all** to verify the NPU drop counters.

Example:

```
Router# show controllers npu stats traps-all instance all location all
Trap Type                                      NPU  Trap  Punt       Punt  Punt  Punt Configured
Hardware   Policer Avg-Pkt Packets              Packets
                                               ID   ID    Dest       VoQ   VLAN  TC   Rate(pps)
Rate(pps)  Level   Size    Accepted             Dropped
=============================================================================================================
L3_ACL_DROP(D)                                 0    106  NPU_DROP    ---   ---   ---  ----------
---------- ---     N/A     0                    0
```

c) Run the **show tech ofa pbr objects** command to verify PBR programming.

Example:

```
Router# show tech ofa pbr objects
ip4pbr element 0 (hdl:0x309826b098):
  base
  |-- dpd_slf -   pending(cr/up/dl):0/0/0, sibling:0x3094b99c78, child:0, num_parents:1, visits:0
```

```
 color_mask:0, last_bwalk_id:0 num_bwalks_started:0
|-- flag - 4000
    |-- flag.is_fwalk_true - 0x1
|-- keylen - 77
|-- trans_id - 88551
|-- create_trans_id - 88499
|-- obj_handle - 0x309826b098
|-- obj_rc - 0x0
|-- reason - 0
|-- table_operation - 6
|-- total_obj_size - 600
|-- idempotent - 0
|-- inflight - 0
|-- table_prop - jid=258 mtime=(UTC)2025.Oct.22 13:36:45.811468
|--  (cont'd)  - replayed=0times
`-- obj_rc - 0:Success
 ofa_npu_mask_t npu_mask => 1
@uint32_t npu_id => 0
@ofa_policymap_name_t policymap_name => pbr_policy:0
 uint32_t pbr_acl_id => 1
@uint32_t ace_seq_num => 0
@uint32_t entry_index => 0
 dpa_ip_addr_t src_ip_addr => 192.1.1.0   <-- VERIFY SOURCE IP
 dpa_ip_mask_t src_ip_mask => 255.255.255.0   <-- VERIFY DESTINATION IP
 dpa_ip_addr_t dest_ip_addr => 201.0.1.1
 dpa_ip_mask_t dest_ip_mask => 255.255.255.0
 uint8_t tcp_flags => 16   VERIFY TCP FLAGS (0x10)
 uint8_t tcp_flags_mask => 255
 dpa_l4_port_t src_port => 1024   <-- VERIFY SOURCE PORT
 dpa_l4_port_mask_t src_port_mask => 65535
 dpa_l4_port_t dest_port => 1024
 dpa_l4_port_mask_t dest_port_mask => 65535
 dpa_port_range_info src_port_range => (not set)
 dpa_port_range_info dest_port_range =>     0/0
 uint8_t proto => 6
 uint8_t proto_mask => 255
 uint8_t flowtag => 10
 uint8_t flowtag_mask => 63
```

The above sample displays only a part of the actual output; the actual output displays more details.